FIG. 1

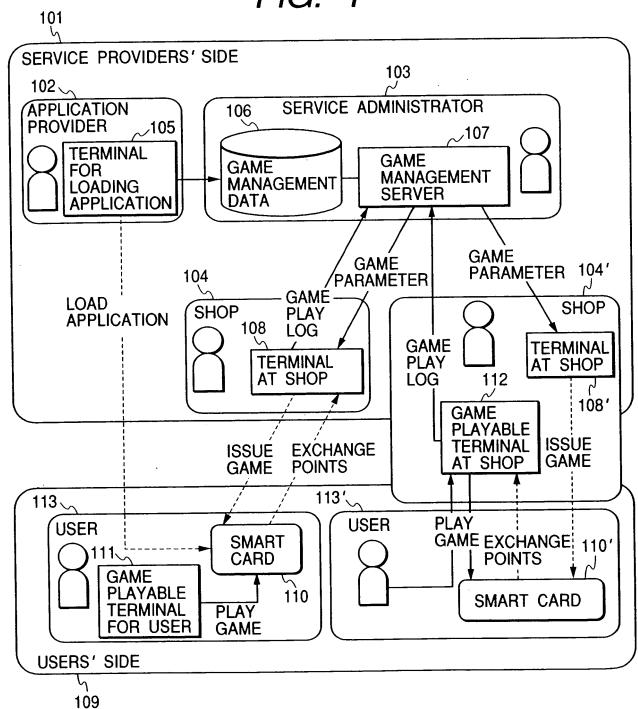
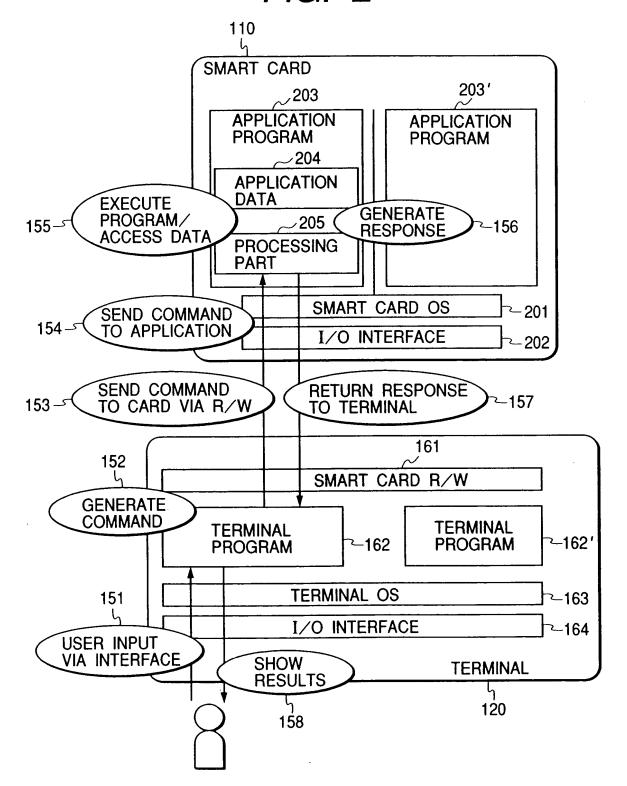


FIG. 2



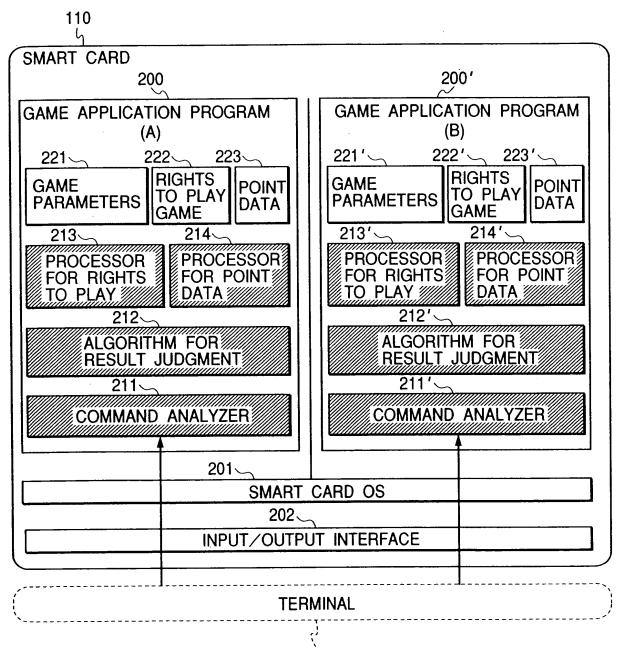


FIG. 4

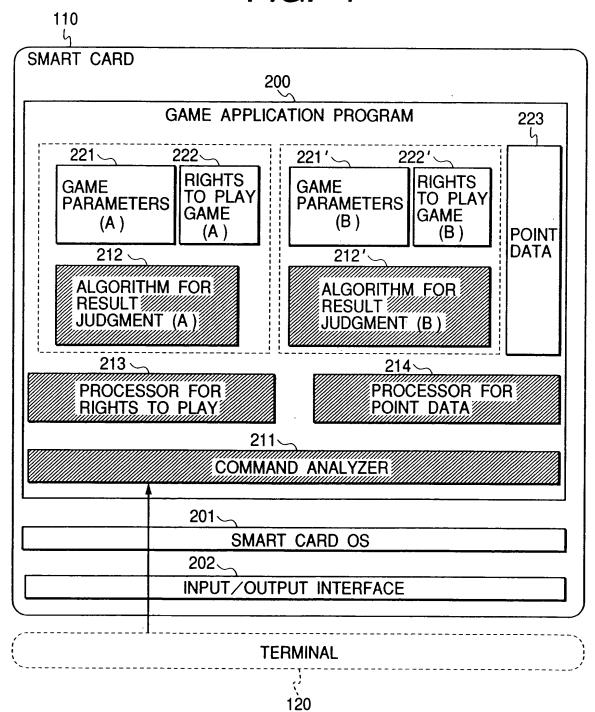


FIG. 5 110 SMART CARD 200 GAME APPLICATION PROGRAM 224 223 221 < 222 < 221′ __ 222' RIGHTS RIGHTS TO GAME **GAME** TO **PLAY PARAMETERS** PLAY **PARAMETERS GAME** COMMON **GAME** (B) (A) **PROCESS** (A) (B) **POINT** COMPONENTS **DATA** 215 215′ GAME (A) MAIN GAME (B) MAIN **DEFINITION OF DEFINITION OF** COMPONENT CALL COMPONENT CALL 225 225' 213 \ 214 PROCESSOR FOR PROCESSOR FOR RIGHTS TO PLAY POINT DATA // 211 COMMAND ANALYZER 201 __ SMART CARD OS 202 INPUT/OUTPUT INTERFACE **TERMINAL**

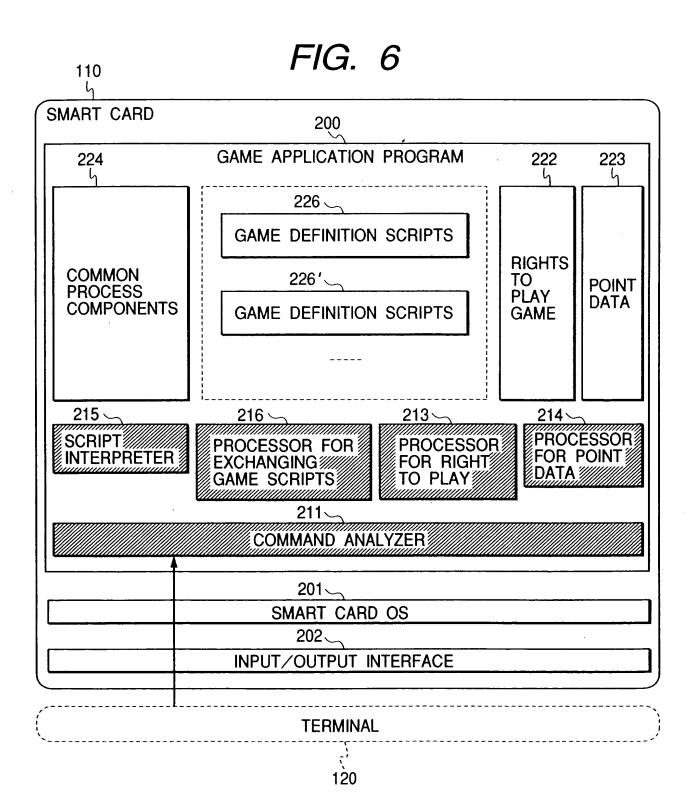
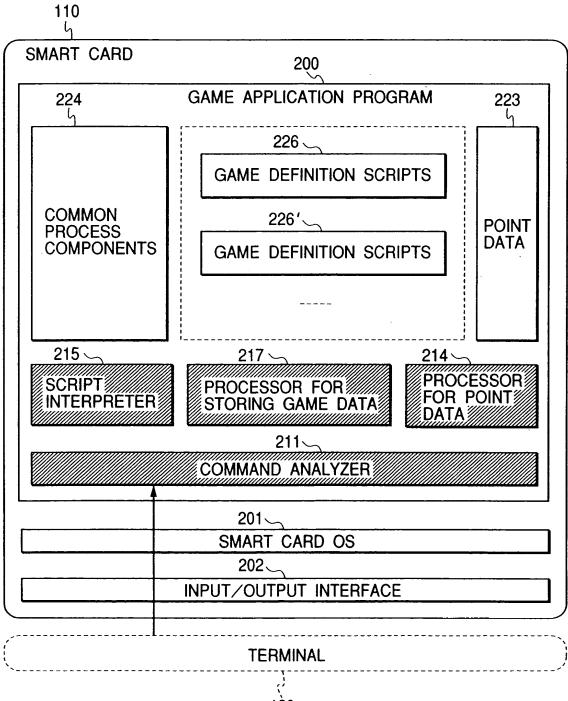
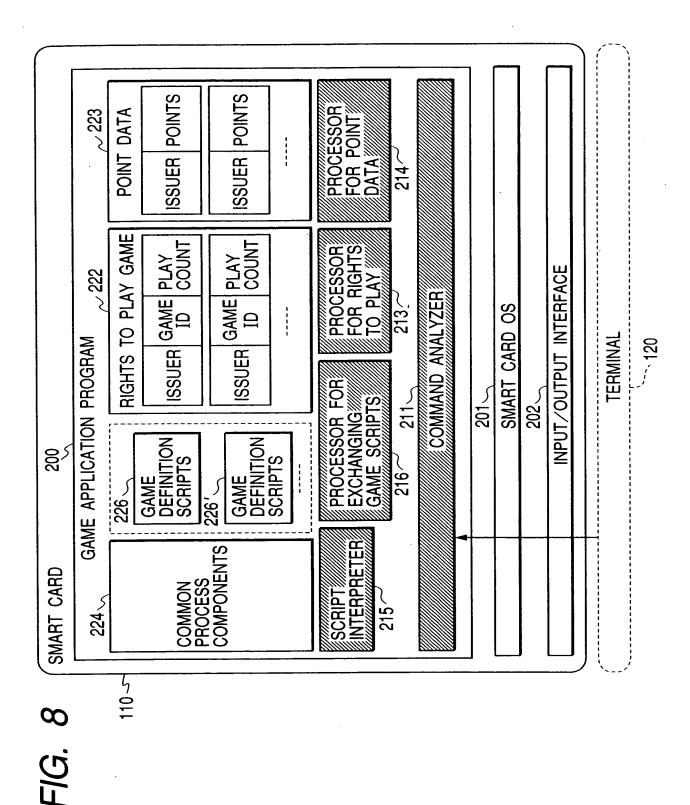
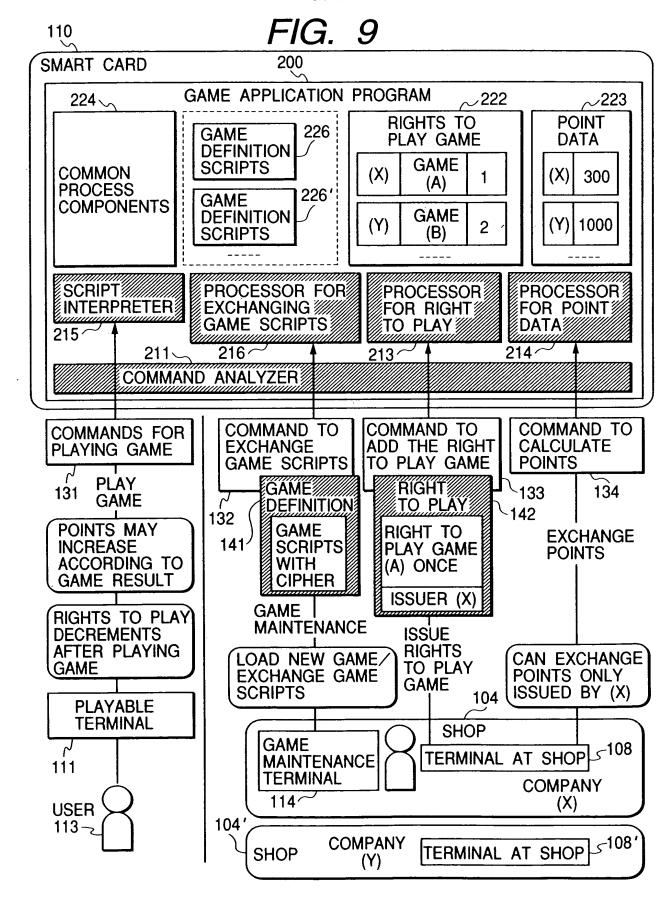
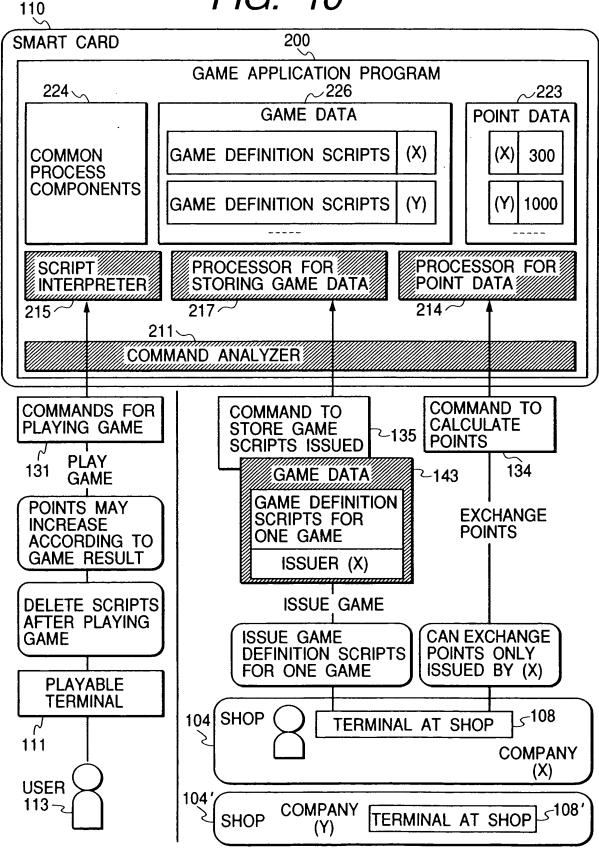


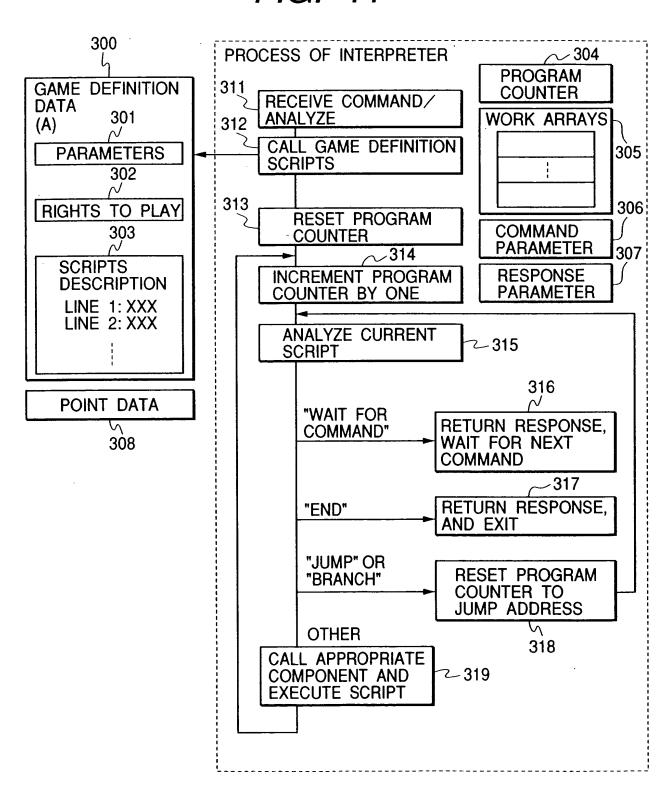
FIG. 7



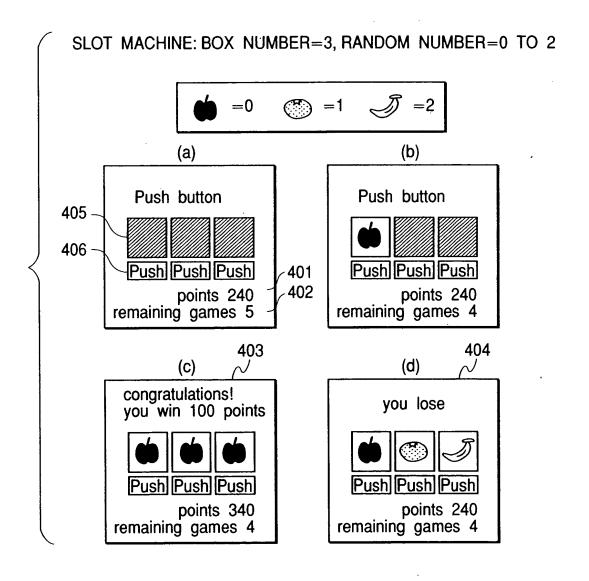






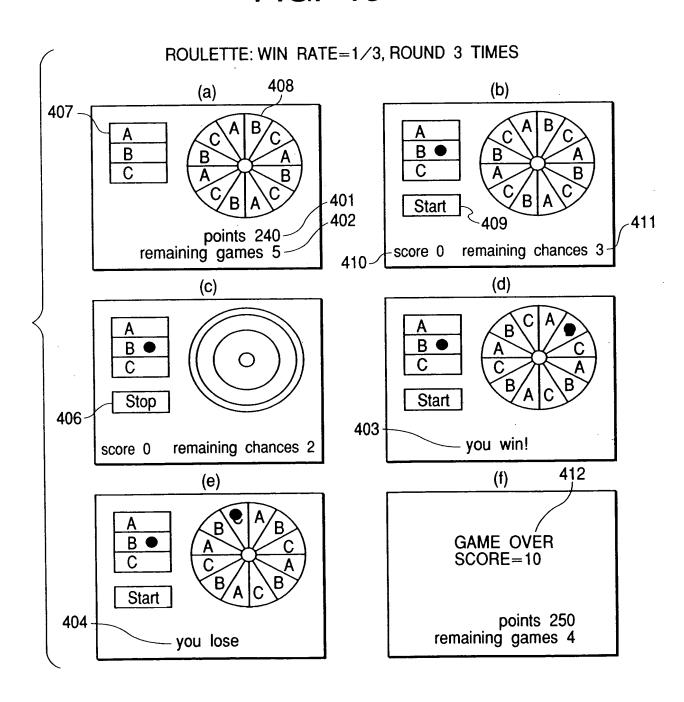


set (a, b) Store value b into array[a] random (a, b) Generate a random number (0 to b) and store it into array[a] Store value of cmd[b] into array[a] setrspa (a, b) Store value a into rsp[b] getcmda (a, b) Getcmda (a, b) Gaddition/subtraction and compare] adda (a, b, c) Add the contents of array[b] and the contents of array[c] and store the result into array[a] suba (a, b, c) Compare array[b] and array[c] from the contents of array[b] and store the result into array[a] cmpa (a, b, c) Compare array[b] and array[c] contents and store the result into array[a] (0×00: equal, 0×01: yy>zz, 0×02: yy <zz) (a)="" (a,="" [point="" [return="" a="" add="" addv="" and="" array[a]="" array[b]="" as="" b="" b,="" c="" c)="" calculation]="" cmpv="" command="" compare="" contents="" data="" data,="" end="" eq="" exit="" for="" from="" game="" if="" into="" jump="" line="" next="" of="" point="" pointa="" pointv="" response="" result="" return="" rsp[]="" script<="" store="" subtract="" td="" terminal]="" the="" to="" value="" wait=""><td colspan="5">cmd[]: command parameter (306)</td></zz)>	cmd[]: command parameter (306)				
suba (a, b, c) Subtract the contents of array[c] from the contents of array[b] and store the result into array[a] cmpa (a, b, c) Compare array[b] and array[c] contents and store the result into array[a] (0×00: equal, 0×01: yy>zz, 0×02: yy <zz) (0×00:="" (a,="" 0×01:="" add="" addv="" and="" array[a]="" array[b]="" b,="" c="" c)="" cmpv="" compare="" contents="" equal,="" from="" into="" of="" result="" store="" subtract="" subv="" the="" value="" yy="">zz, 0×02: yy<zz) (a)="" (a,="" [jump,="" [point="" [return="" a="" add="" and="" array[a]="" as="" b="" b,="" branch]="" c="" c)="" calculation]="" command="" contents="" data="" data,="" end="" eq="" exit="" for="" from="" game="" if="" jmp="" jump="" line="" ne="" next="" of="" point="" pointa="" response="" return="" rsp[]="" script<="" td="" terminal]="" the="" to="" value="" wait=""><td>set (a, b) random (a, b) getcmd (a, b) setrspa (a, b) setrspv (a, b) getcmda (a, b) [addition/subtra</td><td>Generate a random number (0 to b) and store it into array[a] Store value of cmd[b] into array[a] Store the contents of array[a] into rsp[b] Store value a into rsp[b] Store the contents of array[(cmd[b])] into array[a] action and compare] Add the contents of array[b] and the contents of array[c] and</td></zz)></zz)>	set (a, b) random (a, b) getcmd (a, b) setrspa (a, b) setrspv (a, b) getcmda (a, b) [addition/subtra	Generate a random number (0 to b) and store it into array[a] Store value of cmd[b] into array[a] Store the contents of array[a] into rsp[b] Store value a into rsp[b] Store the contents of array[(cmd[b])] into array[a] action and compare] Add the contents of array[b] and the contents of array[c] and			
cmpa (a, b, c) Compare array[b] and array[c] contents and store the result into array[a]	suba (a, b, c)	Subtract the contents of array[c] from the contents of array[b]			
addv (a, b, c) Add the contents of array[b] and the value of c and store the result into array[a] subv (a, b, c) Subtract the value of c from the contents of array[b] and store the result into array[a] cmpv (a, b, c) Compare the contents of array[b] and the value of c and store the result into array[a] (0×00: equal, 0×01: yy>zz, 0×02: yy <zz) (a)="" (a,="" [jump,="" [point="" [return="" a="" add="" and="" array[a]="" array[a]≠="" as="" b="" b,="" branch]="" c="" c)="" calculation]="" command="" contents="" data="" data,="" end="" eq="" exit="" for="" from="" game="" if="" jmp="" jump="" line="" ne="" next="" of="" point="" pointa="" pointv="" response="" return="" rsp[]="" script<="" td="" terminal]="" the="" to="" value="" wait=""><td>cmpa (a, b, c)</td><td>Compare array[b] and array[c] contents and store the result into array[a]</td></zz)>	cmpa (a, b, c)	Compare array[b] and array[c] contents and store the result into array[a]			
subv (a, b, c) Subtract the value of c from the contents of array[b] and store the result into array[a] cmpv (a, b, c) Compare the contents of array[b] and the value of c and store the result into array[a] (0×00: equal, 0×01: yy>zz, 0×02: yy <zz) (a)="" (a,="" [jump,="" [point="" [return="" a="" add="" and="" array[a]="" array[a]≠="" as="" b="" b,="" branch]="" c="" c)="" calculation]="" command="" contents="" data="" data,="" end="" eq="" exit="" for="" from="" game="" if="" jmp="" jump="" line="" ne="" next="" of="" point="" pointa="" pointv="" response="" return="" rsp[]="" script<="" td="" terminal]="" the="" to="" value="" wait=""><td>addv (a, b, c)</td><td>Add the contents of array[b] and the value of c and store the</td></zz)>	addv (a, b, c)	Add the contents of array[b] and the value of c and store the			
store the result into array[a] (0×00: equal, 0×01: yy>zz, 0×02: yy <zz) (a)<="" [jump,="" branch]="" jmp="" td=""><td>subv (a, b, c)</td><td>Subtract the value of c from the contents of array[b] and store the result into array[a]</td></zz)>	subv (a, b, c)	Subtract the value of c from the contents of array[b] and store the result into array[a]			
[jump, branch] jmp (a) Jump to line a eq (a, b, c) Jump to line c if the contents of array[a]= the value of b ne (a, b, c) Jump to line c if the contents of array[a]≠ the value of b [point calculation] pointa (a) Add the contents of array[a] to point data pointv (a) Add value a to point data [return response to terminal] return Return rsp[] as response data, and wait for next command end Return rsp[] as response data, and exit from game script	cmpv (a, b, c)	store the result into array[a]			
jmp (a) Jump to line a eq (a, b, c) Jump to line c if the contents of array[a]= the value of b ne (a, b, c) Jump to line c if the contents of array[a]≠ the value of b [point calculation] pointa (a) Add the contents of array[a] to point data pointv (a) Add value a to point data [return response to terminal] return Return rsp[] as response data, and wait for next command end Return rsp[] as response data, and exit from game script	[iump, branch]	(0, 5, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,			
eq (a, b, c) Jump to line c if the contents of array[a]= the value of b ne (a, b, c) Jump to line c if the contents of array[a]≠ the value of b [point calculation] pointa (a) Add the contents of array[a] to point data pointv (a) Add value a to point data [return response to terminal] return Return rsp[] as response data, and wait for next command end Return rsp[] as response data, and exit from game script	_ ·				
[point calculation] pointa (a) Add the contents of array[a] to point data pointv (a) Add value a to point data [return response to terminal] return Return rsp[] as response data, and wait for next command end Return rsp[] as response data, and exit from game script	eq (a, b, c)	Jump to line c if the contents of array[a] = the value of b			
pointa (a) Add the contents of array[a] to point data pointv (a) Add value a to point data [return response to terminal] return Return rsp[] as response data, and wait for next command end Return rsp[] as response data, and exit from game script					
pointv (a) Add value a to point data [return response to terminal] return Return rsp[] as response data, and wait for next command end Return rsp[] as response data, and exit from game script					
[return response to terminal] return Return rsp[] as response data, and wait for next command end Return rsp[] as response data, and exit from game script	1 '				
return Return rsp[] as response data, and wait for next command end Return rsp[] as response data, and exit from game script					
	return	Return rsp[] as response data, and wait for next command Return rsp[] as response data, and exit from game script			



00 ra	andom (0, 2)	
	11140111 (0, <i>L</i>)	Store a random number (0 to 2) into array[0]
01 se	etrspa (0, 0)	Store the contents of array[0] into rsp[0]
02 re	eturn	Return response, and wait for command
03 ra	andom (1, 2)	Store a random number (0 to 2) into array[1]
04 se	etrspa (1, 0)	Store the contents of array[1] to rsp[0]
05 re	eturn	Return response, and wait for command
06 ra	andom (2, 2)	Store a random number (0 to 2) into array[2]
07 s	etrspa (2, 0)	Store the contents of array[2] into rsp[0]
08 se	etrspv (1, 1)	Store a value of 1 (to indicate the lost) into rsp[1]
09 cı	1 1 1	Compare array[0] and array[1] contents and store
		the result into array[3]
0a ci		Compare array[0] and array[2] contents and store
<u> </u> 		the result into array[4]
0b e	q (3, 0, 0d)	Jump to line 0×0d if array[3] contains 0
0c e	nd	Return response, and exit
0d e	1 \ ' ' '	Jump to line 0×0f if array[4] contains 0
0e e	i e	Return response, and exit
	\ <i>'</i>	Add 50 to point data
10 s	etrspv (2, 50)	Store 50 (points won at this game) into rsp[2]
11 e	nd	Return response, and exit

FIG. 15



UNE 00 01 02	SCRIPT getcmd (0, 0) random (1, 2) cmpa (2, 0, 1)	COMMENT Store the value of cmd[0] into array[0] Store a random number (0 to 2) into array[1] Compare array[0] and array[1] contents and store
03 04 05 06 07 08	setrspa (1, 0) setrspa (2, 1) return getcmd (0, 0) random (1, 2) cmpa (3, 0, 1)	the result into array[2] Store the contents of array[1] into rsp[0] Store the contents of array[2] into rsp[1] Return response, and wait for command Store the value of cmd[0] into array[0] Store a random number (0 to 2) into array[1] Compare array[0] and array[1] contents and store
09 0a 0b	setrspa (1, 0) setrspa (2, 1) return	the result into array[3] Store the contents of array[1] into rsp[0] Store the contents of array[2] into rsp[1] Return response, and wait for command
0c 0d 0e	getcmd (0, 0) random (1, 2) cmpa (4, 0, 1)	Store the value of cmd[0] into array[0] Store a random number (0 to 2) into array[1] Compare array[0] and array[1] contents and store the result into array[4]
0f 10 11 12 13 14 15 16 17 18	setrspa (1, 0) setrspa (2, 1) setrspv (1, 2) ne (2, 0 14) set (5, 20) ne (3, 0, 16) addv (5, 5, 20) ne (4, 0, 16) addv (5, 5, 20) pointv (5) setrspa (5, 2) end	Store the contents of array[1] into rsp[0] Store the contents of array[2] into rsp[1] Store a value of 1 (to indicate the last) into rsp[1] Jump to line 0×14 if the contents of array[2] $\neq0$ Store 20 (points won per game) into array[5] Jump to line 0×16 if the contents of array[3] $\neq0$ Add 20 (points won per game) to array[5] Jump to line 0×18 if the contents of array[4] $\neq0$ Add 20 (points won per game) to array[5] Add the contents of array[5] to points Store the contents of array[5] (final points) into rsp[2] Return response, and exit

SHOOTING GAME: TARGET NUMBER=5, ROUND NUMBER=5

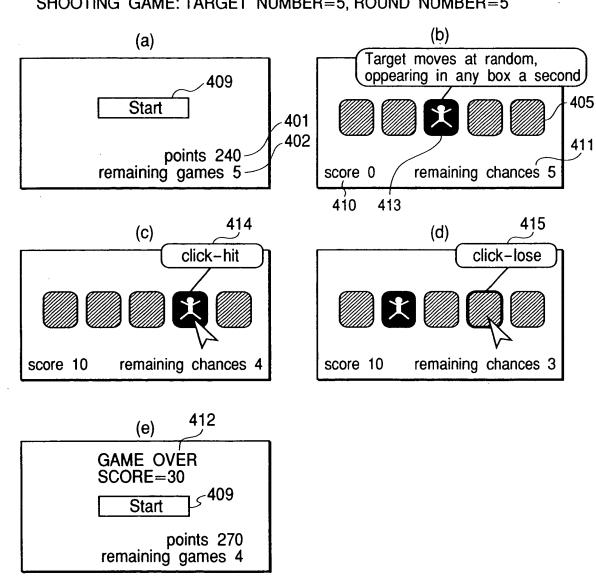
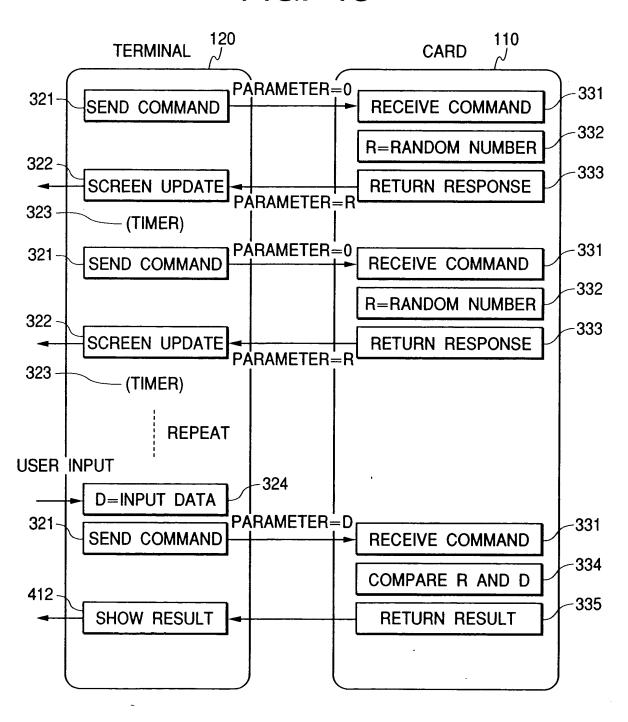


FIG. 18



LINE	SCRIPT	COMMENT
00	set (0, 0)	Store 0 into array[0] (loop index)
01	set (1, 0)	Store 0 into array[1] (points won)
02	set (2, 0,)	Store 0 into array[2] (random number to be
	(, , ,	compared with input)
03	getcmd (3, 0)	Store the value of cmd[0] into array[3]
04	eq (3, 0, 12)	Jump to line 0×12 if array[3] contents 0
		(Comparing values follows)
05	cmpa (4, 2, 3)	Compare array[2] and array[3] contents and store
	, , ,	the result into array[4]
06	setrspa (4, 0)	Store the contents of array[4] (result of comparison) into rsp[0]
07	addv (0, 0, 1)	Increment the counter value contained in array[0]
0/	audv (0, 0, 1)	by one
-08	setrspa (0, 1)	Store the contents of array[0] (loop count) into rsp[1]
09	ne (4, 0, 0b)	Jump to line 0×0 b if the contents of array[4] $\neq 0$
"	110 (4, 0, 00)	(means loosing)
0a	addv (1, 1, 10)	Add 10 (additional points) to array[1]
0b	eq (0, 5, 0e)	Jump to line $0\times0e$ if the contents of array[0]=5 (last)
0c	return	Return response and wait for command
0d	jump (03)	Jump to line 0×03 (beginning of loop)
	, , ,	(Last loop execution follows)
0e	pointv (1)	Add the contents of array[1] to points
Of	setrspa (1, 2)	Store the contents of array[1] (points) into rsp[2]
10	setrspv (1, 3)	Store a value of 1 (to indicate the last) into rsp[3]
11	end	Return response and exit
		(Generating a random number follows)
12	random (2, 5)	Store a random number (0 to 4) into array[2]
13	addv (2, 2, 1)	Increment the value contained in array[2] by one
	•	(the range of random numbers change to 1 to 5)
14	setrspa (2, 0)	Store the contents of array[1] (random number)
		into rsp[0]
15	return	Return response and wait for command
16	jump (0, 3)	Jump to line 0×03 (beginning of loop)

